

AFIT/GOA/ENS/95M-10



AN IMPROVED SOLUTION METHODOLOGY  
FOR THE ARSENAL EXCHANGE MODEL (AEM)

THESIS

Jeffery D. Weir, Captain, USAF

AFIT/GOA/ENS/95M-10

19950503 089

Approved for public release; distribution unlimited

### DISCLAIMER STATEMENT

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification .....	
By .....	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

AFIT/GOA/ENS/95M-10

AN IMPROVED SOLUTION METHODOLOGY FOR THE ARSENAL EXCHANGE  
MODEL (AEM)

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Jeffery D. Weir, B.E.E., MAS

Captain, USAF

MARCH, 1995

Approved for public release; distribution unlimited

## THESIS APPROVAL

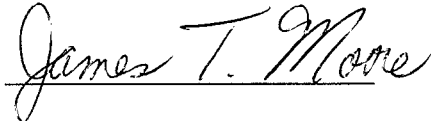
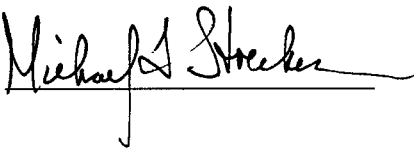
STUDENT: Jeffery D. Weir

CLASS: GOA 95M

THESIS TITLE: An Improved Solution Methodology for the Arsenal Exchange Model  
(AEM)

DEFENSE DATE: 23 February 1995

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
------------	-----------------	-----------

Advisor	LtCol James T. Moore/ENS	
Reader	Capt Michael G. Stoecker/ENC	

## **ACKNOWLEDGMENTS**

I have had a great deal of help in writing this thesis. I would like to thank Mr. Danny C. Myers for his help with the FORTRAN code used in my thesis. A special thanks also to Mr. William L. Cottsworth for answering all my questions about the AEM. I also owe a great deal of thanks to Lt Col James Moore and Capt Michael Stoecker for their guidance and insightful and timely inputs throughout my thesis effort. Most of all, I would like to thank my wife, Cindy and son, Christopher, for their patience, support and encouragement.

Jeffery D. Weir

# Table of Contents

	Page
Acknowledgments .....	ii
List of Figures .....	v
List of Tables.....	vi
Abstract.....	vii
I. Introduction .....	1-1
Background.....	1-1
Inputs .....	1-3
Weapons and Targets .....	1-3
Hedges .....	1-4
Mathematical Model Formulation.....	1-4
Problem Statement .....	1-6
Purpose Statement.....	1-6
Scope .....	1-6
Format .....	1-7
II. Literature Review .....	2-1
Introduction .....	2-1
Goal Programming .....	2-1
Pre-emptive Goal Programming .....	2-4
Goal Programming in the AEM .....	2-5
Goals .....	2-6
AEM Pre-emptive Goal Programming.....	2-7
Alternative Solver Routines .....	2-8
Conclusion .....	2-10

III. Methodology .....	3-1
Introduction .....	3-1
Improved Solution Methodology .....	3-1
Improved Solution Algorithm .....	3-2
Conclusion .....	3-12
IV. Implementation And Testing.....	4-1
Implementation.....	4-1
Testing.....	4-1
Test Case 1 .....	4-1
Test Case 2 .....	4-3
Test Case 3 .....	4-3
Test Case 4 .....	4-5
Test Case 5 .....	4-6
V. Conclusions and Recommendations.....	5-1
Conclusions .....	5-1
Recommendations .....	5-1
Implementation in the AEM .....	5-1
Alternate Optimals .....	5-2
References.....	REF-1
Vita .....	Vita-1

## List of Figures

Figure	Page
3.1 Flow Chart of Improved Solution Algorithm .....	3-9
3.2 Flow Chart of Goal Programming Sequence .....	3-10
3.3 Flow Chart of Pre-emptive Goal Programming .....	3-11



## List of Tables

Table	Page
2.1 Possible Weapon/Target Combinations and Damage Expectancy .....	2-2
2.2 Goals .....	2-2
2.3 Penalty Costs .....	2-3
4.1 Solution Parameters, Test Case 1 .....	4-2
4.2 Solution Vectors, Test Case 1 .....	4-2
4.3 Solution Parameters, Test Case 2 .....	4-3
4.4 Solution Vectors, Test Case 2 .....	4-4
4.5 Solution Parameters, Test Case 3 .....	4-4
4.6 Solution Vectors, Test Case 3 .....	4-5
4.7 Solution Parameters, Test Case 4 .....	4-6
4.8 Solution Vectors, Test Case 4 .....	4-7
4.9 Solution Parameters, Test Case 5 .....	4-8

### **ABSTRACT**

The Arsenal Exchange Model (AEM) allocates weapons to targets using linear programs (LP) formulated by the model. In creating the LPs, the AEM generates weapon constraints, target constraints, and goal or hedge constraints. The current solution method uses a revised simplex algorithm to determine an optimal solution. In order to use this methodology, some of the original constraints are modified. Also, the current algorithm uses a computationally slow matrix inverter which introduces precision error and increases the overall solution time. The improved methodology uses a revised simplex algorithm to first solve a subproblem having only the weapon constraints generated by the AEM. Given this optimal allocation, hedge constraints and target constraints that are violated by the current solution are added to the original subproblem. A dual simplex algorithm is used to find the optimal solution for this new subproblem. By only adding the violated constraints, redundant and identical constraints are not included in any of the subproblems. This eliminates the need to alter the problem as before, and also allows the use of a faster, state of the art, matrix inverter. Additionally, since fewer constraints are used to find the overall optimal solution, fewer computations are necessary. This new methodology was used to solve five test cases. In four of the five test cases, the improved solution methodology produced an optimal integer solution. In all five test cases, it maintained damage expectancy and target coverage, and its solution achieved higher goal satisfaction than the current method.

# **AN IMPROVED SOLUTION METHODOLOGY FOR THE ARSENAL EXCHANGE MODEL (AEM)**

## **I. Introduction**

### **Background**

The Arsenal Exchange Model (AEM) is an aggregated, two-sided, strategic exchange model used primarily for strategic weapon system analysis, strategic nuclear policy and arms control analysis, and general strategic calculations (2:4). It performs allocations of weapons to targets in an attempt to maximize the amount of damage done to targets. Since both the number of weapons and targets are integer values, the AEM formulates the problem as an integer programming (IP) problem. The variables of the problem represent weapon/target combinations that can achieve a desired level of damage expectancy. Often, more than a million variables are considered, and finding an optimal integer solution would require the use of too much time and resource.

Fortunately, an optimal integer solution is not required by the users. The users prefer a good feasible integer solution quickly. Currently this is accomplished by solving the linear programming (LP) relaxation of the IP and using a heuristic to find a feasible integer solution (6). There are some problems, however, with the current LP solver used by the AEM.

In creating the LP, the AEM generates weapon constraints, target constraints, and goal or hedge constraints based on user specified inputs. The AEM solver currently uses a combination of column generation, generalized upper bounding (GUB), decomposition,

Gauss-Jordan elimination, and compact tableau storage. Like all column generating techniques, the AEM solves a sequence of LPs. The AEM solves the original LP by first solving an LP with a subset of the strategies. When that subset has been solved to optimality, the AEM's column generator uses the basis inverse matrix of that solution to generate strategies that will improve the current objective function value. These columns are then incorporated into a new LP with the basis matrix, and the AEM solves this new LP to optimality. The AEM continues this iterative process until all possible columns have been generated, or until no new columns will increase the value of the current objective function. For large, complex AEM applications, precision and speed problems are noticeable (5:881). The AEM solver uses several techniques to alleviate the precision problems. Among these are re-inversion and backward pivoting. Although these techniques alleviate a large portion of the precision problems, those caused by redundant or identical constraints are not entirely corrected.

The hedge constraints are used to ensure the solution meets certain force employment policies (2:30). Thus, hedge constraints are used to incorporate goals into the model. At certain points during the solution procedure, the hedge constraints can be identical to other constraints already generated by the AEM. These identical or redundant constraints cause singularity problems in the Gauss-Jordan elimination algorithm (see Chapter II). Since singular matrices cannot be inverted, the AEM multiplies the right hand side of the hedge constraints by 0.9995 (3). The new matrix, while nearly singular, can now be inverted. While the adjustment factor does eliminate the singularity problem, it adds to the precision problem in two ways. First, it changes the problem being solved. The LP will now solve a problem that ensures only 99.95% of force employment policies are met. Second, the near singularity of the new matrix causes mathematical precision problems during the Gauss-Jordan pivots. In addition, the Gauss-Jordan inverter algorithm also has speed problems.

The Gauss-Jordan inverter algorithm is used because of its robustness. Although the Gauss-Jordan routine can invert the matrix, it is not as efficient as other routines (9:25). That is, the robustness of a Gauss-Jordan inverter is at the expense of efficiency. Modern, state-of-the-art routines recognize the near singularity of the matrix and will not invert it (7:31). If a technique can be found to eliminate the identical constraints, then the correction factor can be eliminated and a more precise and efficient inversion routine can be used.

## **Model Description**

This section gives a brief description of the AEM. It describes the user inputs to the model, and how the mathematical model is formulated.

**Inputs.** There are two basic types of user inputs for the AEM. The first type consists of the resources available to each side. These include the weapons to be used and the targets to be attacked. The second type, a set of user controls known as hedges, set goals and requirements for the allocation of the resources.

**Weapons and Targets.** The user is allowed to define up to 100 weapon types (2:39). Some examples of weapon types are B-52s, Minuteman IIIs, and Trident D-5s. The user inputs the number of weapons of each type, as well as a number of other weapon characteristics, such as, the number of warheads delivered by each weapon, circular error probable (CEP), reliability, and availability. Similarly, the target set is broken up into classes. The limit on the number of target classes is 2500 minus the number of weapon types (2:39). The AEM assumes each weapon type is seen by the other side as a target class. Within each target class, the user sets the number of targets in that class and inputs a variety of target characteristics including hardness, size, and value.

**Hedges.** Hedge constraints are input by the user to ensure the AEM meets certain force employment policies. The user also determines how the AEM will use these

inputs. The input can be used as a goal to be attained or as a requirement that must be met. In the first, the user does not associate a priority with the goal, and the AEM tries to minimize all of the deviations from the user's goals through goal programming techniques (see Chapter II). Failure to meet a goal affects the goodness of the allocation. In the second, the user specifies a priority with the requirement, and the AEM attempts to satisfy the user specified requirements in priority order using pre-emptive goal programming techniques (see Chapter II). In this case, failure to meet a requirement affects the feasibility of the allocation.

**Mathematical Model Formulation.** The mathematical model formulated by the AEM is a linear program (LP), which is shown in Equations (1-1) to (1-4). In the LP, the objective function (1-1) maximizes damage expectancy subject to three sets of constraints. The first set of constraints (1-2) ensures that the number of weapons used from each weapon class does not exceed the number of weapons available in that class. The second set of constraints (1-3) ensures that the total number of targets attacked in each target class does not exceed the total number of targets in that class. Finally, the third set of constraints (1-4) specifies user input hedges. The hedge constraints are added to the LP using goal programming (see Chapter II). After formulating the model, the AEM uses a column generating algorithm to solve the LP.

Each column in the LP is a strategy. A strategy is a weapon/target combination that achieves a certain damage expectancy. For example if two warheads of weapon type 1 can be used against one target in target class A with a damage expectancy of 0.73, the strategy column in the LP would look like:

$\frac{0.73 * \text{STRAT}_j}{2 * \text{STRAT}_j}$	Objective function
0	Weapon constraint 1
$\vdots$	
0	
$\frac{1 * \text{STRAT}_j}{1 * \text{STRAT}_j}$	Target constraint A
0	
$\vdots$	
0	

where  $\text{STRAT}_j$  is the variable representing the number of times weapon/target combination  $j$  is used, and  $\text{DE}_j = 0.73$ ,  $W_{jl} = 2$ , and  $T_{jA} = 1$ . The entire LP formulation then has the following mathematical representation:

$$\text{Max} \quad \sum_j \text{STRAT}_j * \text{DE}_j - \sum_g M_g * \text{DIFF}_g \quad (1-1)$$

$$\text{Subject To} \quad \sum_j W_{jl} * \text{STRAT}_j \leq \text{WEP}_l \quad \text{for } l = 1, \dots, o \quad (1-2)$$

$$\sum_j \text{STRAT}_j * T_{jk} \leq \text{TAR}_k \quad \text{for } k = 1, \dots, p \quad (1-3)$$

$$\sum_j \text{Hedge}_g (...) + \text{DIFF}_g = \text{HEG}_{0g} \quad \text{for } g = 1, \dots, q \quad (1-4)$$

$$\text{STRAT}_j \geq 0 \quad \text{for } j = 1, \dots, n$$

Where:

$m$  = the total number of constraints =  $o + p + q$

$n$  = the number of allowable strategies

$o$  = the number of weapon types

$p$  = the number of target classes

$q$  = the number of hedges

$\text{STRAT}_j$  = the number of times strategy  $j$  is chosen

$DE_j$	= the damage expectancy of strategy $j$
$M_g$	= the penalty amount for not achieving hedge $g$
$DIFF_g$	= the amount by which hedge $g$ was not achieved
$W_{jl}$	= the number of type $l$ weapons used in strategy $j$
$WEP_l$	= the total number of type $l$ weapons
$T_{jk}$	= the number of type $k$ targets attacked in strategy $j$
$TAR_k$	= the total number of type $k$ targets
$HEG_{og}$	= the level to be achieved by hedge $g$
$Hedge_g(...)$	= a linear function of the previously defined variables

### **Problem Statement**

The AEM suffers from numerical and precision problems that increase solution time and degrade the possibility to meet policy specified goals (5:881).

### **Purpose Statement**

The purpose of this research is to improve the LP solution methodology in the AEM. This new methodology should be computationally faster and contain less precision error than the current methodology.

### **Scope**

The two most important factors associated with running the AEM are solution time and precision. To determine if the improved methodology is better than the current one, these two parameters are measured. Solution time is defined as the time required to solve to optimality all of the LPs generated by the AEM. Time is measured in CPU seconds. Precision is measured by comparing the damage expectancy and goal performance of the continuous solutions found by each methodology.



## **Format**

Chapter II provides an explanation of goal programming in general and its implementation in the AEM. Chapter II also presents an explanation of solution techniques that have been used to overcome problems similar to those encountered by the AEM. The improved solution methodology is presented in Chapter III. Chapter IV discusses implementation and test results. Chapter V presents conclusions and recommendations.

## **II. Literature Review**

### **Introduction**

This chapter first gives a brief description of goal programming, followed by a discussion of the AEM's implementation of goal programming. Then it examines some techniques to eliminate the problems associated with the current LP solution methodology.

### **Goal Programming**

Goal programming is one technique that might be used when a decision maker faces multiple objectives, or goals. In general, LPs have only one objective function. Goal programming is one method that can be used to treat the multiple objectives in a fashion which permits the formulation of the problem as a linear programming model. There are two ways of implementing goal programming. In both methods, deviation variables are used to convert each objective into a constraint for the LP (11:189).

The first method assigns a penalty per unit deviation from each objective and uses linear programming to minimize the total penalty incurred because of unmet objectives. This is illustrated in the following example.

---

**Example 2.1**      A decision maker has to determine the optimal use of weapons against a group of targets while meeting fiscal and military objectives. There are 10 weapons and two groups of 5 targets each. Table 2.1 outlines the possible weapon/target combinations and levels of damage, and Table 2.2 lists the goals of the decision maker.

NOTE: All costs are in thousands of dollars

---

**Table 2.1 Possible Weapon/Target Combinations and Damage Expectancy**

Strategy Number	# of Weapons Needed	# and type of Targets Attacked	Damage Expectancy	Total Cost of Weapons
STRAT <sub>1</sub>	3	1 type A	0.6	\$30
STRAT <sub>2</sub>	2	1 type B	0.6	\$20
STRAT <sub>3</sub>	1	1 type B	0.3	\$10

**Table 2.2 Goals**

Goal #	Description
1	Spend less than \$100
2	Attack at least 80% of target class A
3	Attack at least 60 % of target class B

Given the information in Table 2.1, the decision maker could base his objective function on either the cost of, or the damage expectancy of, attacking the two target sets. For an objective function minimizing the cost to destroy the target sets, the LP formulation is

$$\begin{aligned}
 \text{Min} \quad & 30x_1 + 20x_2 + 10x_3 \\
 \text{Subject To} \quad & 3x_1 + 2x_2 + 1x_3 \leq 10 \\
 & x_1 \leq 5 \\
 & x_2 + x_3 \leq 5 \\
 & 30x_1 + 20x_2 + 10x_3 \leq 100 \quad (\text{Goal 1}) \\
 & x_1 \geq 4 \quad (\text{Goal 2}) \\
 & x_2 + x_3 \geq 3 \quad (\text{Goal 3}) \\
 & x_1, x_2, x_3 \geq 0
 \end{aligned} \tag{2-1}$$

where  $x_i$  is the number of times STRAT <sub>$i$</sub>  is employed.

Regardless of the objective function, there is no point that will satisfy the weapon and target constraints and satisfy all three goals. The decision maker must now determine the penalties for not satisfying a goal. Table 2.3 presents these penalties.

**Table 2.3 Penalty Costs**

Goal #	Penalty Cost
1	\$1 per unit short
2	\$60 per unit short
3	\$20 per unit short

The new LP can be formulated to minimize the cost of deviating from the three goals. Each of the goal constraints in (2-1) must now be converted into equality constraints with variables which represent a deviation from the goals. Since it is not known if the solution will undersatisfy or oversatisfy a given goal, the deviation variables are defined in the following manner:

$s_i^+$  = amount by which the solution numerically exceeds goal  $i$

$s_i^-$  = amount by which the solution is numerically under goal  $i$

The new LP can be written as:

$$\begin{aligned}
 \text{Min} \quad & 30x_1 + 20x_2 + 10x_3 + 1s_1^+ + 60s_2^- + 20s_3^- \\
 \text{Subject To} \quad & 3x_1 + 2x_2 + 1x_3 \leq 10 \\
 & x_1 \leq 5 \\
 & x_2 + x_3 \leq 5 \\
 & 30x_1 + 20x_2 + 10x_3 + s_1^- - s_1^+ = 100 \quad (\text{Goal 1}) \\
 & x_1 + s_2^- - s_2^+ = 4 \quad (\text{Goal 2}) \\
 & x_2 + x_3 + s_3^- - s_3^+ = 3 \quad (\text{Goal 3}) \\
 & x_1, x_2, x_3, s_i^+, s_i^- \geq 0 \quad \text{for } i = 1, 2, 3
 \end{aligned} \tag{2-2}$$

An optimal solution to (2-2) is  $x_1 = 3, x_2 = 0, x_3 = 1, s_1^- = 0, s_2^- = 1, s_3^- = 2, s_1^+ = 0, s_2^+ = 0, s_3^+ = 0$ , with an objective function value of 200. The first goal can be achieved, but the second and third goals are not achieved as the solution will only be able to attack three of the type A targets and one of the type B targets. This method only works, however, if the penalties for failing to reach each goal are known. Often, the penalty costs cannot be determined precisely, but it may be possible to rank order the goals according to importance. In this case, pre-emptive goal programming can be used.

**Pre-emptive Goal Programming.** In pre-emptive goal programming, the goals are ranked from most important to least important. In the objective function, the penalty coefficient for each goal is  $P_j$ , where it is assumed that for  $n$  goals (11:178)

$$P_1 \gg P_2 \gg \dots \gg P_n$$

Using this notation, the pre-emptive goal programming formulation of Example 2.1 is

$$\begin{aligned}
 \text{Min} \quad & 30x_1 + 20x_2 + 10x_3 + P_1s_1^+ + P_2s_2^- + P_3s_3^- \\
 \text{Subject To} \quad & 3x_1 + 2x_2 + 1x_3 \leq 10 \\
 & x_1 \leq 5 \\
 & x_2 + x_3 \leq 5 \\
 & 30x_1 + 20x_2 + 10x_3 + s_1^- - s_1^+ = 100 \quad (\text{Goal 1}) \\
 & x_1 + s_2^- - s_2^+ = 4 \quad (\text{Goal 2}) \\
 & x_2 + x_3 + s_3^- - s_3^+ = 3 \quad (\text{Goal 3}) \\
 & x_1, x_2, x_3, s_i^+, s_i^- \geq 0 \quad \text{for } i = 1, 2, 3
 \end{aligned} \tag{2-3}$$

There are two approaches for solving (2-3). The first approach is to use the goal programming simplex algorithm (11:179). In practice, however, the  $P_i$ 's would cause computational problems and are usually not used. Instead, the pre-emptive goal programming approach uses the fact that higher priority goals must be optimized before lower priority goals can be considered (10:203). One way to accomplish this is to solve a series of problems where the objective function of each problem's linear programming

model is the objective function which only includes the deviation variables associated with the highest-priority goal not yet optimized. Applying this approach to Example 2.1, and assuming the goals listed in Table 2.3 are in order of importance, the first LP solved is

$$\begin{aligned}
 \text{Min} \quad & 30x_1 + 20x_2 + 10x_3 + s_1^+ \\
 \text{Subject To} \quad & 3x_1 + 2x_2 + 1x_3 \leq 10 \\
 & x_1 \leq 5 \\
 & x_2 + x_3 \leq 5 \\
 & 30x_1 + 20x_2 + 10x_3 + s_1^- - s_1^+ = 100 \quad (\text{Goal 1}) \\
 & x_1 + s_2^- - s_2^+ = 4 \quad (\text{Goal 2}) \\
 & x_2 + x_3 + s_3^- - s_3^+ = 3 \quad (\text{Goal 3}) \\
 & x_1, x_2, x_3, s_i^+, s_i^- \geq 0 \quad \text{for } i = 1, 2, 3
 \end{aligned} \tag{2-4}$$

From the solution to (2-4)  $s_1^+ = 0$ . Therefore, goal 1 is satisfied. The next LP would minimize the deviation from the second goal, and would also contain a constraint to ensure that the first goal attainment is not changed. Its formulation is

$$\begin{aligned}
 \text{Min} \quad & 30x_1 + 20x_2 + 10x_3 + s_2^- \\
 \text{Subject To} \quad & 3x_1 + 2x_2 + 1x_3 \leq 10 \\
 & x_1 \leq 5 \\
 & x_2 + x_3 \leq 5 \\
 & 30x_1 + 20x_2 + 10x_3 + s_1^- - s_1^+ = 100 \quad (\text{Goal 1}) \\
 & x_1 + s_2^- - s_2^+ = 4 \quad (\text{Goal 2}) \\
 & x_2 + x_3 + s_3^- - s_3^+ = 3 \quad (\text{Goal 3}) \\
 & s_1^+ = 0 \\
 & x_1, x_2, x_3, s_i^+, s_i^- \geq 0 \quad \text{for } i = 1, 2, 3
 \end{aligned} \tag{2-5}$$

This process continues until all of the goals have been included in the model.

## Goal Programming in the AEM

This section covers the aspects of goal programming that are unique to the AEM. First, it details the types of goals used by the AEM and explains how the AEM currently

solves the problem of duplicate rows in the constraint matrix. Then it outlines the current method used by the AEM to implement pre-emptive goal programming.

**Goals.** The goals used by the AEM are all user specified. These goals are input as hedges. There are five categories of hedges (2:144).

1. Requirements pertaining to the total damage to a specified set of target classes with a specified set of weapon types. This hedge type is known as a "value hedge". An example of this is the value destroyed on target class A by weapon type 7 must be greater than 0.7 times the total value of targets in target class A.
2. Requirements pertaining to the total number of weapon re-entry vehicles (RVs) allocated against a specified set of target classes by a specified set of weapon types. This hedge type is known as a "weapon hedge". An example of this is the number of RVs from weapon types 2 and 3 on target classes A and D must equal 1000.
3. Requirements pertaining to the total number of targets within a specified set of target classes hit by a specified set of weapon types. This is known as a "target hedge". An example of this is cover all targets in class B with weapon types 2 or 3.
4. Requirements pertaining to how much damage must be inflicted on each target within the specified target class. This is known as a "CDEMIN hedge" (CDEMIN is the minimum allowable damage expectancy). An example of this is the CDEMIN for all targets in class A by an RV from any weapon type must equal 0.8.
5. Requirements that all strategies for a specified weapon/target combination must satisfy a specified set of criteria. The set of criteria can specify the amount of damage obtained by the strategy, the number of RVs used in the strategy, the weapon types used in a strategy, or the presence of at least two legs of the triad in the strategy.

In many cases, the user defined goals are similar to constraints already generated by the AEM. For example, a user goal might be to cover all targets in class A.

Mathematically, this hedge constraint would look like:

$$\sum_j \text{STRAT}_j * T_{jA} + \text{DIFF}_I = \text{TAR}_A \quad (2-6)$$

where  $\text{STRAT}_j$  is the number of times strategy  $j$  is chosen,  $T_{jA}$  is the number of targets in class A attacked by strategy  $j$ ,  $\text{DIFF}_I$  is the deviation variable and represents the targets in class A not attacked, and  $\text{TAR}_A$  is the number of targets in target class A. The target constraint generated by the AEM for target class A is:

$$\sum_j \text{STRAT}_j * T_{jA} + \text{Slack}_A = \text{TAR}_A \quad (2-7)$$

where  $\text{STRAT}_j$  is the number of times strategy  $j$  is chosen,  $T_{jA}$  is the number of targets in class A attacked by strategy  $j$ ,  $\text{Slack}_A$  is the slack variable associated with this constraint, and  $\text{TAR}_A$  is the number of targets in target class A. If the goal is met, both  $\text{DIFF}_I$  and  $\text{Slack}_A$  will be zero, and the AEM will remove those columns from the original tableau. Once the columns are removed, both Equations (2-6) and (2-7) are identical. This results in a singular basis matrix which is not invertible. To prevent this, the AEM multiplies the right-hand-side of all hedge constraints by 0.9995 (3). This causes the goals and constraints to differ by a small amount. The resulting nearly singular matrices are invertible by some matrix inverters such as Gauss-Jordan Elimination.

**AEM Pre-emptive Goal Programming.** The AEM implements goal programming by solving a series of LPs. The first LP contains all of the weapon and target constraints generated by the AEM and the set of hedge constraints with the highest priority. Once this LP has been solved to optimality, the deviation variable associated with each goal is examined. If the deviation variable for a goal is zero, the goal was satisfied and the column associated with that goal's deviation variable is removed from the tableau. If the deviation variable is non-zero, the right-hand-side of that goal is adjusted to make the deviation variable zero. Once this adjustment has been made, the columns associated with these deviation variables are removed from the tableau. The removal of



the columns of the deviation variables ensures that the solutions to subsequent LPs will not deviate from the level of goal achievement reached in previous LPs. This process is continued until all of the user defined goals have been incorporated in the problem.

### **Alternative Solver Routines**

There are several alternative techniques that can be used to eliminate the problems associated with redundant constraints. These techniques range from combinatorial comparisons of constraints to dual simplex algorithms using constraint selection rules. The simplest technique is to compare each constraint with all linear combinations of the other constraints and delete any that are redundant. For large problems, this can be computationally expensive. If the problem contains  $m$  constraints, the total number of possible linear combinations is given by  $\sum_{i=2}^m \frac{m!}{(m-i)! i!}$ , which simplifies to  $2^m - m - 1$ .

For a problem with only 30 constraints, this is over a billion comparisons. Although this approach would eliminate the precision problems, it greatly increases the time needed to solve the original LP. Insight into the specific problem may provide information to determine how the redundant constraints are being generated.

In the AEM, the hedge constraints can be redundant. If the deviation variables are not removed from the basis, and constraints setting these deviation variables to the best value obtained in previous LPs are added to the problem, redundancy can be eliminated. Since there will be no redundancy, use of an adjustment factor is not necessary. Although precision would be increased by eliminating the correction factor, solving time could also be increased due to the potential introduction of a large number of new constraints. The speed problem caused by these new constraints could be overcome by using more efficient inverter algorithms such as LU factorization. The efficiency of these algorithms should at least maintain current solution times (9:37-38). A solution technique which eliminated the

redundant constraints while making the original LP smaller would decrease the solution time and increase precision.

One way to make the original problem smaller is to note that for most LPs only a small percentage of the constraints are binding at optimality (8:177). Therefore, most of the computational effort can be avoided if these binding constraints are identified before running the solver routine. Although the redundant constraints may be binding, only one instance of the constraint is needed. After solving a portion of the problem which does not contain redundant constraints, a selection algorithm can be used to iteratively select groups of constraints that are violated by the current solution. These violated constraints can then be added to the current subproblem so a new subproblem is created. This new subproblem can be solved quickly using a dual simplex algorithm. For example, given the following LP, there are several constraint selection rules that can be used.

$$\begin{array}{ll} \text{MAX} & c'x \\ \text{s. t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

where  $c$  and  $x$  are elements of  $R^n$ ,  $b$  is an element of  $R^m$  and strictly greater than 0, and  $A$  is an  $m \times n$  matrix of coefficients  $a_{ij}$ . Let  $V_k = \{ i : \sum_{j=1}^n a_{ij} * x_j^k > b_i \}$  where  $x_j^k$  is the  $j^{\text{th}}$  element of the solution vector  $x$  at iteration  $k$ . Thus,  $V_k$  is the index set of violated constraints at iteration  $k$ . Three of the most common violated constraint selection rules (8:178) are described below.

1. *Largest Summation Rule (LS)*. Divide each constraint by its right-hand-side to transform the constraint into the form  $\sum_{j=1}^n \bar{a}_{ij} x_j \leq 1$ , where  $\bar{a}_{ij}$  equals  $a_{ij}$  divided by  $b_i$ . For each  $i \in V_k$ , let

$$S_i = \sum_{j=1}^n |\bar{a}_{ij}|$$

and calculate

$$S_{m1} = \max_{i \in V_k} \{S_i\} \quad \text{and} \quad S_{m2} = \max_{i \in V_k, i \neq m1} \{S_i\}$$

Select those constraints with indices  $m1$  and  $m2$  to add to the current problem at each iteration.

2. *Most Violated Constraint Rule (MV)*. For each  $i \in V_k$ , select the constraints that yield the two largest values for  $\sum_{j=1}^n a_{ij} * x_j^k - b_i$ . Since at the first iteration, these values cannot be calculated, the first two constraints must be chosen randomly.
3. *Random Selection Rule (RS)*. Two constraints are selected at random from  $V_k$ . Note this rule will not work for the AEM as it might choose a redundant constraint and maintain the current precision problem.

Using these selection rules and a dual simplex algorithm, Myers attained, on average, a 90% reduction in the time required to solve randomly generated linear programming problems that were  $300 \times 500$  and  $300 \times 320$  (8).

## Conclusion

Since the AEM performs pre-emptive goal programming in its solution process and since adding two constraints at a time could add redundant constraints, Myers algorithm cannot be directly implemented in the AEM. It does, however, suggest a solution approach for improving the AEM. The solution methodology developed in this thesis is similar to Myers in that it uses a constraint selection algorithm and dual simplex pivots to update the solution at each iteration. It, however, also incorporates the goal programming of the AEM and uses algorithms resident in the AEM. Chapter III presents the improved solution methodology and its algorithm.

### **III. Methodology**

#### **Introduction**

In this chapter, the improved solution methodology is developed. Then, the model formulation and solution procedure are presented.

#### **Improved Solution Methodology**

Although the current solution methodology does solve the allocation problem, there is room for improvement. The first area where improvement is needed is the required adjustment of the right-hand side of all hedge constraints. The deletion of this requirement increases the precision of the solution and allows the use of a more efficient matrix inverter. As mentioned before, this can be done if any redundant hedge constraints can be identified and not included in the LP. The second area which would improve the model is reduction of the size of the LP. This can be done in two ways. First, redundant hedge constraints can be eliminated. Second, non-binding constraints can be eliminated. Redundant constraints occur when hedge inputs are the same as the AEM generated constraints. A non-binding constraint would be any constraint that, if removed from the LP, would not affect the optimal solution. To identify the redundant and non-binding constraints, the improved methodology first solves a subproblem of the original LP.

By solving a subproblem consisting only of the weapon constraints (3-2), a bounded super-optimal solution to the original LP (3-1) can be found. The weapon constraints were chosen for two reasons. First, the AEM is designed to allocate all input weapons. If weapons are not used, a warning is given to the user. For this reason, users commonly install a target that is easy to attack and results in a very high payoff. This "sink" will have weapons allocated to it that are not necessary to satisfy the goals the user has specified. Second, the weapon constraints guarantee a bounded solution since every strategy must contain a usable weapon. Once the super-optimal solution is found using a

revised simplex algorithm, the remaining constraints of (3-1) can be checked for violations. After it has been determined which constraints are violated, they must be incorporated into the LP. This creates an updated LP. A dual simplex algorithm can be used to find a new feasible, optimal solution to the updated problem. With this new solution, the remaining constraints can be checked again to determine if any are violated. This iterative procedure continues until no constraints are violated by the current solution.

Determining which violated constraints to incorporate at each iteration of this process is difficult. Myers suggests incorporating two constraints at each iteration (8). However, this could add identical constraints to the problem. Testing revealed adding only one violated constraint to the problem at each iteration resulted in a large number of dual simplex pivots and iterations. To decrease the number of pivots and iterations, and ensure that identical constraints are not added to the problem, up to two violated non-hedge constraints can be added, or a violated hedge constraint is added by itself. When both non-hedge and hedge constraints are violated, only a hedge constraint is added.

### **Improved Solution Algorithm**

This section presents the step-by-step algorithm used to implement the improved solution methodology. First, the general AEM formulation presented in Chapter I is shown. This is followed by a presentation of the algorithm. Figures 3.1 through 3.3 show the algorithm's flowchart.

The original LP as formulated by the AEM and discussed in Chapter I is written below.

$$\begin{aligned}
 \text{Max} \quad & \sum_j \text{STRAT}_j * \text{DE}_j - \sum_g M_g * \text{DIFF}_g \\
 \text{Subject To} \quad & \sum_j W_{jl} * \text{STRAT}_j \leq \text{WEP}_l \quad \text{for } l = 1, \dots, o \\
 & \sum_j \text{STRAT}_j * T_{jk} \leq \text{TAR}_k \quad \text{for } k = 1, \dots, p \\
 & \sum_j \text{Hedge}_g(\dots) + \text{DIFF}_g = \text{HEG}_{og} \quad \text{for } g = 1, \dots, q \\
 & \text{STRAT}_j \geq 0 \quad \text{for } j = 1, \dots, n
 \end{aligned} \tag{3-1}$$

Where:

- m = the total number of constraints = o + p + q
- n = the number of allowable strategies
- o = the number of weapon types
- p = the number of target classes
- q = the number of hedges
- STRAT<sub>j</sub> = the number of times strategy j is chosen
- DE<sub>j</sub> = the damage expectancy of strategy j
- M<sub>g</sub> = the penalty amount for not achieving hedge g
- DIFF<sub>g</sub> = the amount by which hedge g was not achieved
- W<sub>jl</sub> = the number of type l weapons used in strategy j
- WEP<sub>l</sub> = the total number of type l weapons
- T<sub>jk</sub> = the number of type k targets attacked in strategy j
- TAR<sub>k</sub> = the total number of type k targets
- HEG<sub>og</sub> = the level to be achieved by hedge g
- Hedge<sub>g</sub>(...) = a linear function of the previously defined variables

The improved algorithm is presented below.

STEP 1. Obtain columns for problem (3-1) from the AEM. Initialize the sets  
 $G = \{\text{indices for hedge constraints. } 1, 2, \dots, q\}$

$K = \{\text{indices for non - hedge constraints. } 1, 2, \dots, p\}$ . Let  $r$  be the number of times (3-2) is updated and solved. Initialize  $r$  to 1. If no columns exist, skip to STEP 15.

STEP 2. Solve problem (3-2), using a revised simplex algorithm (1:294).

$$\begin{aligned} \text{Max} \quad & \sum_j \text{STRAT}_j * \text{DE}_j \\ \text{Subject To} \quad & \sum_j W_{jl} * \text{STRAT}_j \leq \text{WEP}_l \quad \text{for } l = 1, 2, \dots, o \\ & \text{STRAT}_j \geq 0 \quad \text{for } j = 1, 2, \dots, n \end{aligned} \quad (3-2)$$

Let  $\text{STRAT}^r$  be the solution vector obtained.

STEP 3. Determine the type of goal programming required to solve user specified goals. If pre-emptive goal programming is required, skip to STEP 8.

STEP 4. Generate the set

$$V^r = \left\{ \begin{array}{l} i: \sum_{j=1}^n T_{ij} * \text{STRAT}_j^r > \text{TAR}_i \quad \text{for } i \in K, \text{ and} \\ i: \sum_{j=1}^n \text{Hedge}_i(\dots) * \text{STRAT}_j^r \neq \text{HEG}_{oi} \quad \text{for } i \in G \end{array} \right\}$$

where  $\text{STRAT}_j^r$  is the  $j^{\text{th}}$  element of the solution vector at iteration  $r$ .

Thus,  $V^r$  is the index set of violated constraints at iteration  $r$  and  $|V^r| = \alpha$ . If  $\alpha = 0$ , then the solution is optimal so return to STEP 1.

Otherwise continue.

STEP 5. If  $V_i^r \in G$ , for  $i = 1, 2, \dots, \alpha$ , determine which violated hedge constraint to add to (3-2) using the largest summation selection rule, remove the

index of this constraint from the set  $G$ , set  $r = r + 1$ , and skip to STEP 7.

STEP 6. No hedge constraints are violated, so  $V_i^r \in K$ , for  $i = 1, 2, \dots, \alpha$ . If  $\alpha = 1$ , add constraint  $V_1^r$  to (3-2) and remove this index from the set  $K$ . Otherwise add two violated non-hedge constraints to (3-2) using the largest summation selection rule, and remove the indices of the added constraints from the set  $K$ . Set  $r = r + 1$ .

STEP 7. Solve the updated problem (3-2) using a dual simplex algorithm to obtain  $\text{STRAT}^r$  (1:329). Return to STEP 4.

STEP 8. Initialize  $s$  to 1, where  $s$  is the priority level to be optimized.

STEP 9. Generate the set

$$V^r = \left\{ \begin{array}{l} i: \sum_{j=1}^n T_{ij} * \text{STRAT}_j^r > \text{TAR}_i \text{ for } i \in K, \text{ and} \\ i: \sum_{j=1}^n \text{Hedge}_i(\dots) * \text{STRAT}_j^r \neq \text{HEG}_{oi} \text{ for } i \in G \end{array} \right\}$$

where  $\text{STRAT}_j^r$  is the  $j^{\text{th}}$  element of the solution vector at iteration  $r$ .

Thus,  $V^r$  is the index set of violated constraints at iteration  $r$  and  $|V^r| = \alpha$ . If  $\alpha = 0$ , then the solution is optimal so return to STEP 1.

Otherwise continue.

STEP 10. Determine  $s$ .

STEP 11. For each  $V_i^r \in G$ ,  $i = 1, 2, \dots, \alpha$ , determine the priority of the constraint.

For the first violated hedge constraint  $V_i^r$  with priority  $s$ , update problem (3-2), remove the index of this hedge from the set  $G$ , set  $r = r + 1$ , and skip to STEP 14.

STEP 12. No hedge constraints of the current priority level are violated. If  $V_i^r \in K$ , for  $i = 1, 2, \dots, \alpha$ , determine which violated non-hedge constraints to add to (3-2) using the largest summation selection rule.



- Update problem (3-2) with two violated non-hedge constraints if available, otherwise use one, remove the indices of these constraints from the set  $K$ , set  $r = r + 1$ , and skip to STEP 14.
- STEP 13. For each  $V_i' \in G$ ,  $i = 1, 2, \dots, \alpha$ , the priority is not  $s$  and  $V_i' \notin K$ . Thus, no constraints can be added to (3-2). Examine the solution of (3-2). For all priority  $s$  hedges in (3-2) with non-zero deviation variables, the right-hand-side is adjusted until the deviation variable associated with them is zero. Next, delete the column associated with each deviation variable of a hedge with priority  $s$ , and pivot each deviation variable out of the basis as necessary, set  $s = s + 1$ , and return to STEP 10.
- STEP 14. Solve the new subproblem using a dual simplex algorithm to obtain STRAT'. Return to STEP 9.
- STEP 15. Current solution is the optimal solution for original LP. Produce output and stop.

STEP1 ties the LP solution algorithm to the column generator (refer to Chapter I) in the AEM. At STEP 2, a problem consisting of only the weapon constraints is solved. Before the improved solution methodology can continue, the type of goal programming to implement, standard or pre-emptive, must be determined. This is done at STEP 3. If standard goal programming is to be implemented, the algorithm proceeds to STEP 4. Here, a set consisting of the indices of the constraints that are violated by the current solution is generated. If the set is empty, the solution to the LP is optimal, and the algorithm returns to STEP 1 and adds all newly generated columns the LP. If the set is not empty, the algorithm determines if the set contains the index of a hedge constraint. If hedge constraints are violated, the problem at STEP 2 is updated with the constraint with the largest summation (see Chapter II) at STEP 5, and at STEP 7 a dual simplex algorithm finds the optimal solution of this updated problem. If no hedge constraint was violated,

then at STEP 6 the set must contain the indices of non-hedge constraints. In this case, the problem is updated with two of the non-hedge constraints using the largest summation rule, unless only one non-hedge constraint is violated, and again STEP 7 solves the updated problem. The solution obtained at STEP 7 is then used to generate a new set containing the indices of violated constraints not yet incorporated in problem (3-2). STEPs 4 through 7 are repeated until all constraints have been incorporated in problem (3-2) or until no more constraints are violated. If pre-emptive goal programming was indicated at STEP 3, STEPs 8 through 14 would have been implemented.

At STEP 8, the priority level to be optimized is initialized to 1. STEP 9 is similar to STEP 4. It generates the set containing the indices of violated constraints. Again, if the set is empty, the solution is optimal, and the algorithm returns to STEP 1 where new columns are generated. If the set is not empty, STEP 10 determines the value of the priority to be optimized at this point in the solution process. STEP 11 determines if the set of violated constraints contains the index of a hedge constraint with the same priority value as in STEP 10. It updates the problem with the first violated hedge constraint it finds with this value, and goes to STEP 14 to solve the updated problem using a dual simplex algorithm. If no such hedge is found, STEP 12 checks for any violated non-hedge constraints. It determines the summation values for each violated non-hedge constraint. If only one non-hedge constraint is violated it incorporates that constraint in problem (3-2); otherwise, it updates problem (3-2) with two of the violated non-hedge constraints using the largest summation selection rule. A solution to this new problem is found at STEP 14. STEP 13 is used when the set of violated constraints contains neither a hedge constraint with the current priority value nor any non-hedge constraints. If this happens, the right-hand side of any hedge constraints in problem (3-2) with the current priority level and a non-zero deviation variable are adjusted until the deviation variable associated with the constraint is zero. Next, the columns associated with the deviation variables of the hedges with the current priority level are deleted. Deviation variables that were basic are

pivoted out of the basis. Finally, the priority value is incremented by one, and the algorithm returns to STEP 10. Whenever a new solution is found at STEP 14, the algorithm returns to STEP 9 to determine if there are violated constraints. Finally, when there are no more columns to be added to the LP at STEP 1, STEP 15 produces the output of the optimal solution to the original LP.

Figure 3.1 represents the flow for STEPs 1 through 3 and STEP 15 of the improved solution algorithm. Node 1 is the input to the standard goal programming algorithm in Figure 3.2, while node 2 is the output from that algorithm. Nodes 3 and 4 are the input and output of the pre-emptive goal programming algorithm shown in Figure 3.3.

**Figure 3.1 Flow Chart of Improved Solution Algorithm**

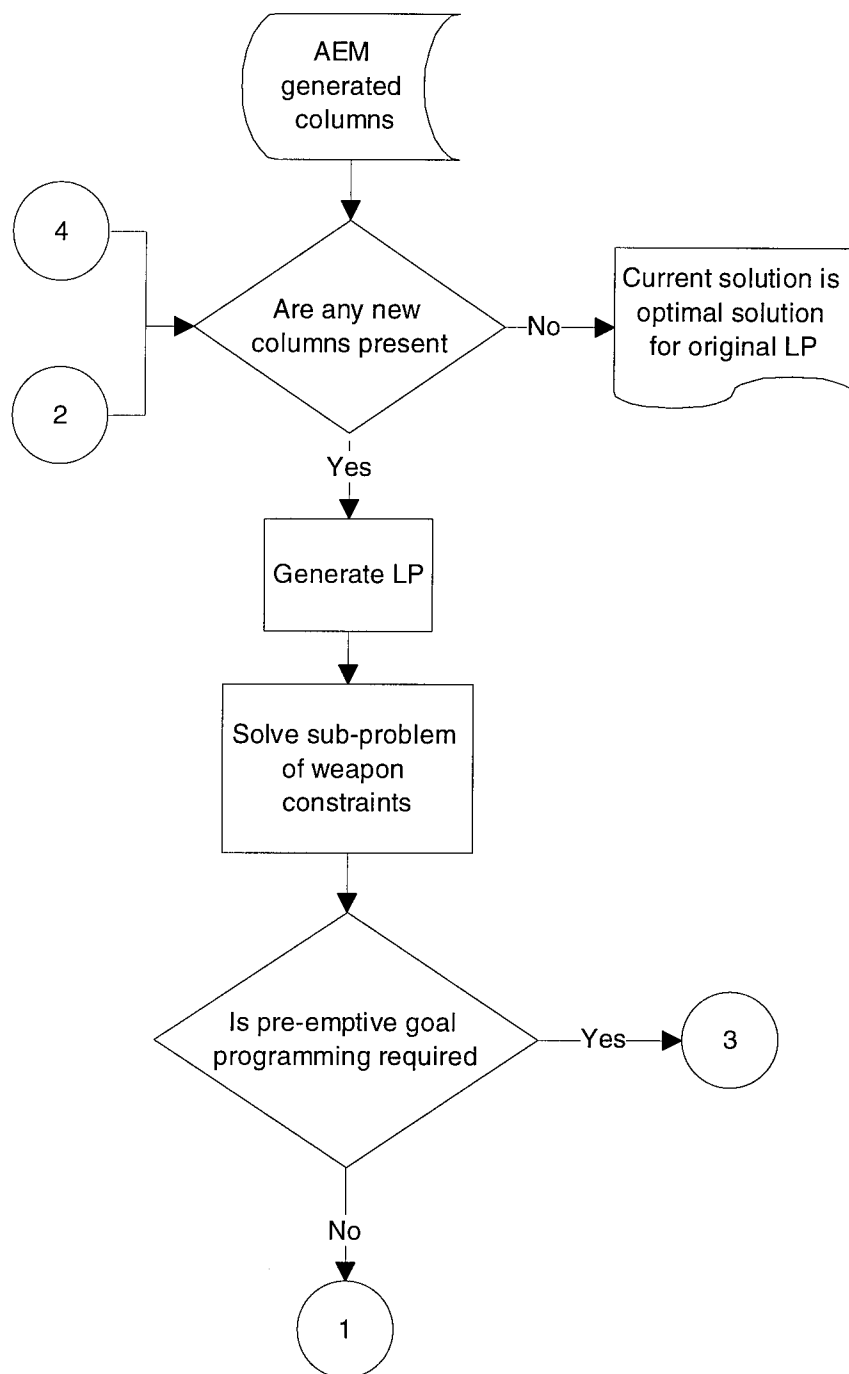


Figure 3.2 shows the flow of STEPs 4 through 7 of the improved solution algorithm.

**Figure 3.2 Flow Chart of Goal Programming Sequence**

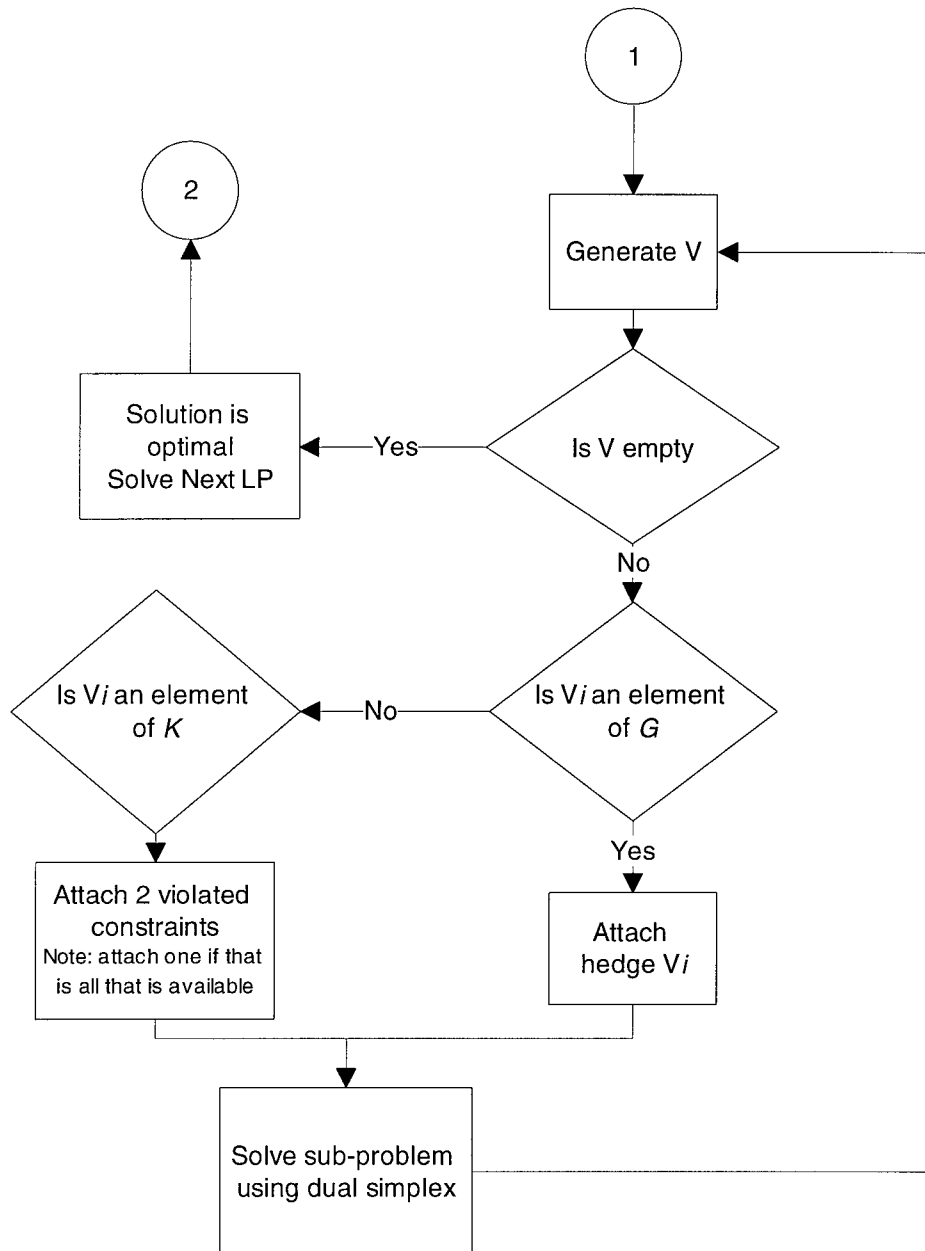
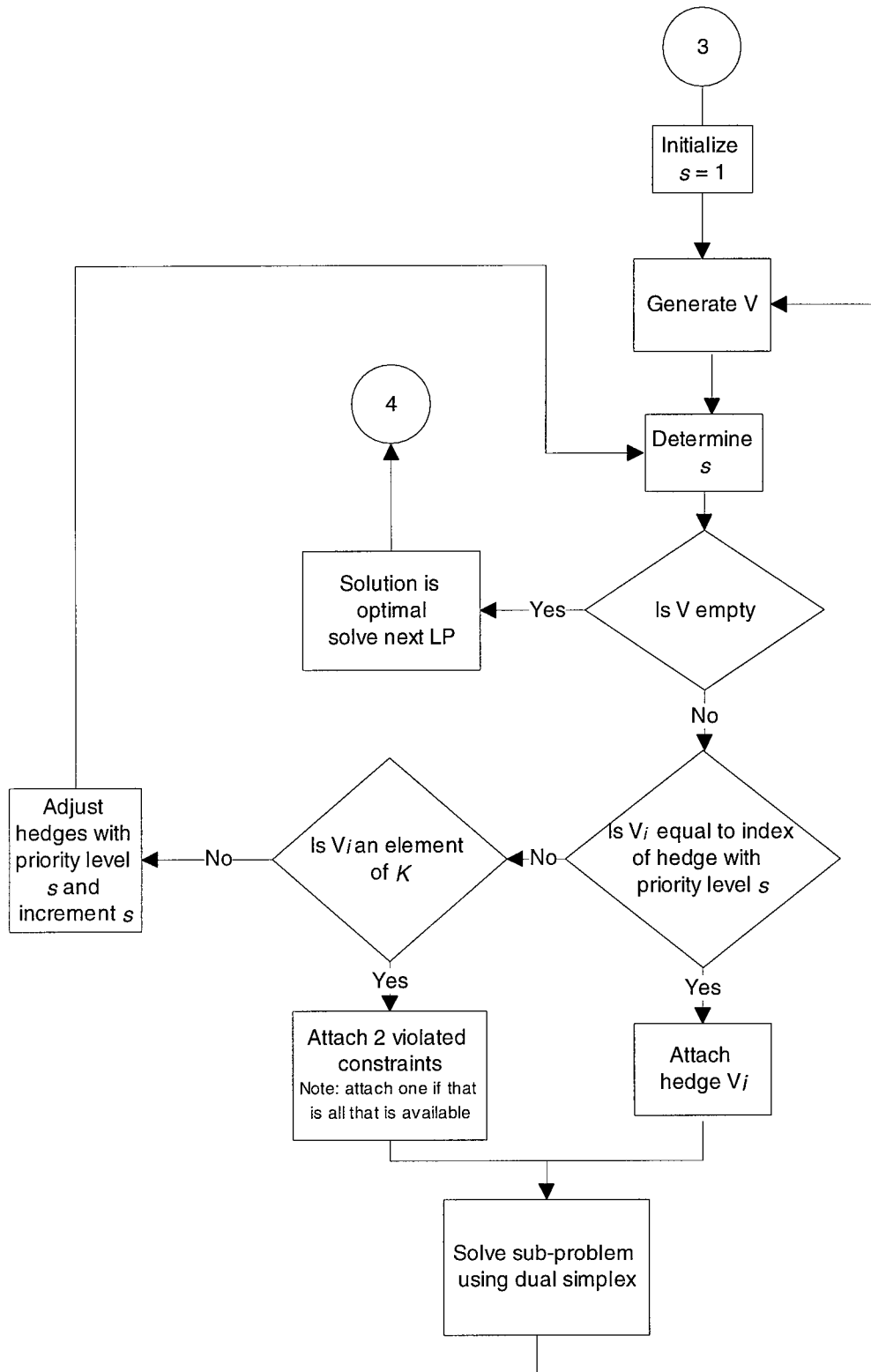


Figure 3.3 shows the flow of STEPs 8 through 14 of the improved solution algorithm..

**Figure 3.3 Flow Chart of Pre-emptive Goal Programming**



## **Conclusion**

The improved solution methodology should decrease the size of the problem solved in order to find the optimal solution of the original LP. It will also solve the original problem without adjusting the hedge constraints with the 0.9995 correction factor. By decreasing the size of the problem and removing the correction factor, this methodology should decrease overall solution time while improving the precision of the solution. Chapter IV describes the implementation and testing of the improved solution algorithm.

## IV. Implementation And Testing

### Implementation

The improved solution methodology (ISM) presented in Chapter III was coded in FORTRAN using double precision mathematics. The revised simplex and dual simplex algorithms are from Best and Ritter (1). The basis matrix was inverted using the standard IMSL real matrix inverter. The IMSL inverter, subroutine name DLINRG , uses LU factorization to compute the inverse of a real square matrix (7:31). The subroutine DLINRG will not invert nearly singular matrices. To determine if a matrix is nearly singular, DLINRG approximates the condition number of the matrix. DLINRG uses the fact that ill conditioned matrices, i.e. nearly singular matrices, have a very large condition number (4:265). If the condition number is greater than  $1/\epsilon$  (where  $\epsilon$  is machine precision), DLINRG outputs an error and the matrix is not inverted (7:31).

### Testing

The methodology was tested on five test cases provided by Mr. William L. Cottsworth, President of AEM Services Incorporated. For each test case, two solutions were produced. Solution 1 is the continuous solution found by the AEM. Solution 2 is the continuous solution found by the improved methodology.

**Test Case 1.** The first test case had 6610 weapons divided into six weapon types and 5388 targets divided into 11 target classes. There were two target hedges specified in a priority order. Table 4.1 shows the results from the application of the two methods. The very slight decrease in Goal 2 performance experienced by the improved solution methodology can be attributed to the higher priority goal being satisfied at a higher level by the improved methodology. Table 4.2 shows the solution vectors from both solvers.



While the solution produced by the AEM was continuous, the improved solution methodology produced an optimal integer solution.

**Table 4.1 Solution Parameters, Test Case 1**

Parameter	Solution 1	Solution 2
Damage Expectancy	0.5664	0.5667
Value Destroyed	5411.84	5414.54
Targets Attacked	4375	4288
Goal 1 Performance	99.95%	100%
Goal 2 Performance	35.08%	35.07%
Solution Time (CPU secs)	$1.5 \times 10^{-6}$	$3.5 \times 10^{-9}$

**Table 4.2 Solution Vectors, Test Case 1**

AEM Strategies	Times Strategy Chosen	Value Attacked	AEM Value Destroyed	Improved Solution Strategies	Times Strategy Chosen	Value Attacked	ISM Value Destroyed
1	384.00	768.00	613.79	1	372.00	744.00	594.60
2	752.10	1504.20	676.14	2	752.00	1504.00	676.05
3	136.00	272.00	217.38	3	148.00	296.00	236.56
4	197.90	197.90	89.06	4	198.00	198.00	89.10
5	36.00	180.00	159.02	5	36.00	180.00	159.02
6	314.00	628.00	383.96	6	314.00	628.00	383.96
7	649.00	1298.00	793.60	7	352.00	704.00	430.43
8	501.90	1003.80	612.82	8	154.00	308.00	188.03
9	506.00	1012.00	770.50	9	482.00	964.00	733.96
10	187.00	374.00	267.62	10	472.00	944.00	675.48
11	108.00	324.00	258.81	11	108.00	324.00	258.81
12	12.00	96.00	88.23	12	0.00	0.00	0.00
13	4.00	296.00	293.59	13	0.00	0.00	0.00
14	500.00	500.00	134.85	14	500.00	500.00	134.85
15	0.10	0.10	0.03	15	0.00	0.00	0.00
16	87.00	87.00	52.46	16	0.00	0.00	0.00
37	0.00	0.00	0.00	37	4.00	296.00	293.60
42	0.00	0.00	0.00	42	12.00	96.00	90.53
65	0.00	0.00	0.00	65	384.00	768.00	469.56
Totals	4375.00	8541.00	5411.84		4288.00	8454.00	5414.54

**Test Case 2.** The second test case had 6610 weapons divided into 6 weapon types and 5388 targets divided into 11 target classes. There were five target hedges specified in a priority order. Table 4.3 shows the results from the application of the two methods. Table 4.4 shows the solution vectors from both solvers. The AEM solution was better only in one category, Goal 5 Performance. The higher goal achievement for Goal 5 can be attributed to the AEM solution not meeting previous goals at the same level as the improved solution. While the solution produced by the AEM was continuous, the improved solution methodology produced an optimal integer solution again.

**Table 4.3 Solution Parameters, Test Case 2**

Parameter	Solution 1	Solution 2
Damage Expectancy	0.5246	0.5246
Value Destroyed	5012.30	5012.44
Targets Attacked	5388	5388
Goal 1 Performance	100%	100%
Goal 2 Performance	99.9%	100%
Goal 3 Performance	99.95%	100%
Goal 4 Performance	99.95	100%
Goal 5 Performance	72.42%	72.39%
Solution Time (CPU secs)	$3.2 \times 10^{-5}$	$7.3 \times 10^{-7}$

**Test Case 3.** The third test case had 6610 weapons divided into 6 weapon types and 5388 targets divided into 11 target classes. There were four target hedges and one weapon hedge specified in a priority order. Table 4.5 shows the results from the two methods. Table 4.6 shows the solution vectors from both solvers. Again, the AEM solution had a better damage expectancy for Goal 5 at the expense of not meeting higher priority goals. The improved solution methodology also produced an optimal integer solution for this test case.

**Table 4.4 Solution Vectors, Test Case 2**

AEM Strategies	Times Strategy Chosen	Value Attacked	AEM (DE) Value Destroyed	Improved Solution Strategies	Times Strategy Chosen	Value Attacked	ISM Value Destroyed
1	3.30	6.60	5.60	1	3.00	6.00	5.09
2	384.00	768.00	613.79	2	384.00	768.00	613.79
3	1552.60	3105.20	837.47	3	1552.00	3104.00	837.15
4	256.00	512.00	409.19	4	256.00	512.00	409.19
5	197.90	197.90	53.43	5	198.00	198.00	53.46
6	99.90	99.90	18.41	6	100.00	100.00	18.43
	999.50	999.50	198.50	7	1000.00	1000.00	198.60
8	0.50	0.50	0.22	8	0.00	0.00	0.00
9	0.10	0.10	0.04	9	0.00	0.00	0.00
10	4.00	296.00	293.38	10	4.00	296.00	293.38
11	36.00	180.00	152.82	11	36.00	180.00	152.82
12	0.00	0.00	0.00	12	0.00	0.00	0.00
13	199.40	398.80	243.83	13	200.00	400.00	244.56
14	474.70	949.40	661.68	14	361.00	722.00	503.20
15	204.10	408.20	310.79	15	205.00	410.00	312.16
16	241.30	482.60	367.44	16	355.00	710.00	540.57
17	114.60	229.20	174.51	17	0.00	0.00	0.00
18	108.00	324.00	275.07	18	108.00	324.00	275.07
19	12.00	96.00	90.37	19	12.00	96.00	90.37
20	500.00	500.00	305.70	20	500.00	500.00	305.70
21	0.10	0.10	0.06	21	0.00	0.00	0.00
61	0.00	0.00	0.00	61	114.00	228.00	158.90
Totals	5388.00	9554.00	5012.30		5388.00	9554.00	5012.44

**Table 4.5 Solution Parameters, Test Case 3**

Parameter	Solution 1	Solution 2
Damage Expectancy	0.5397	0.5398
Value Destroyed	5156.26	5157.40
Targets Attacked	5388	5388
Goal 1 Performance	100%	100%
Goal 2 Performance	99.9%	100%
Goal 3 Performance	99.95%	100%
Goal 4 Performance	99.95%	100%
Goal 5 Performance	49.11%	49.07
Solution Time (CPU secs)	$5.7 \times 10^{-5}$	$1.6 \times 10^{-7}$

**Table 4.6 Solution Vectors, Test Case 3**

AEM Strategies	Times Strategy Chosen	Value Attacked	AEM Value Destroyed	Improved Solution Strategies	Times Strategy Chosen	Value Attacked	ISM Value Destroyed
1	264.00	528.00	421.98	1	372.00	744.00	594.60
2	1052.90	2105.80	567.93	2	1052.00	2104.00	567.45
3	256.00	512.00	409.19	3	148.00	296.00	236.56
4	99.90	99.90	18.41	4	100.00	100.00	18.43
5	500.80	500.80	99.46	5	500.00	500.00	99.30
6	0.10	0.10	0.04	6	0.00	0.00	0.00
7	0.10	0.10	0.05	7	0.00	0.00	0.00
8	197.90	197.90	53.43	8	198.00	198.00	53.46
9	500.00	500.00	255.75	9	0.00	0.00	0.00
10	12.00	96.00	84.81	10	12.00	96.00	84.81
11	36.00	180.00	152.82	11	36.00	180.00	152.82
12	0.00	0.00	0.00	12	0.00	0.00	0.00
13	163.10	326.20	227.35	13	0.00	0.00	0.00
14	150.90	301.80	184.52	14	314.00	628.00	383.96
15	311.90	623.80	434.76	15	225.00	450.00	313.63
1	224.30	448.60	229.46	16	226.00	452.00	231.20
17	827.10	1654.20	1011.38	17	664.00	1328.00	811.94
18	179.80	359.60	273.79	18	373.00	746.00	567.98
19	108.00	324.00	258.94	19	0.00	0.00	0.00
20	4.00	296.00	293.60	20	4.00	296.00	293.60
21	499.20	499.20	178.59	21	500.00	500.00	178.88
23	0.00	0.00	0.00	23	56.00	112.00	85.27
32	0.00	0.00	0.00	32	108.00	324.00	258.81
51	0.00	0.00	0.00	51	500.00	500.00	224.75
Totals	5388.00	9554.00	5156.26		5388.00	9554.00	5157.46

**Test Case 4.** The fourth test case had 6610 weapons divided into 6 weapon types and 5388 targets divided into 11 target classes. There were three target hedges, one weapon hedge, and one value hedge specified in a priority order. Table 4.7 shows the results from applying the two methods. Table 4.8 shows the solution vectors from both solvers. Again Goal 5 performance was lower for the improved solution methodology due

to higher priority goals being satisfied at a higher level than the current solution. The improved solution methodology did not produce an optimal integer solution for this test case, but it did provide almost twice as many integer valued variables in the optimal solution.

**Table 4.7 Solution Parameters, Test Case 4**

Parameter	Solution 1	Solution 2
Damage Expectancy	0.5406	.5406
Value Destroyed	5164.44	5164.67
Targets Attacked	4887.9	4891.41
Goal 1 Performance	100%	100%
Goal 2 Performance	99.95%	100%
Goal 3 Performance	99.95%	100%
Goal 4 Performance	99.95%	100%
Goal 5 Performance	1.05%	0.75%
Solution Time (CPU secs)	$4.3 \times 10^{-7}$	$2.08 \times 10^{-9}$

**Test Case 5.** The fifth test case had 8691 weapons divided into 34 weapon types and 5774 targets divided into 122 target classes. There were a total of 33 hedges listed in a priority order. Table 4.9 shows the results from the application of the two methodologies. Due to the large size of the solution vectors, they are not included in this section. However, the improved solution methodology again produced an optimal integer solution. The improved solution had the same or better goal satisfaction in 28 of the 33 goals. Out of the first 23 goals the improved solution only had two goals with a lower satisfaction level. That is for Goal 4, the current solution attacked 1442.7 targets, while the improved solution attacked 1442 targets. Likewise, for Goal 5 the current solution attacked 1718 targets and the improved solution attacked 1717 targets. The other three goals had a lower level of satisfaction in the improved solution due the limiting of weapons and targets caused by the improved solution's higher goal satisfaction of higher level goals.

**Table 4.8 Solution Vectors, Test Case 4**

AEM Strategies	Times Strategy Chosen	Value Attacked	AEM Value Destroyed	Improved Solution Strategies	Times Strategy Chosen	Value Attacked	ISM Value Destroyed
1	240.00	480.00	383.62	1	240.00	480.00	383.62
2	256.00	512.00	409.19	2	256.00	512.00	409.19
3	69.70	69.70	62.29	3	67.41	67.41	60.24
4	500.80	500.80	99.46	4	500.00	500.00	99.30
5	197.90	197.90	53.43	5	198.00	198.00	53.46
6	4.00	296.00	293.60	6	4.00	296.00	293.60
7	36.00	180.00	152.82	7	36.00	180.00	152.82
8	0.00	0.00	0.00	8	0.00	0.00	0.00
9	108.00	324.00	258.94	9	108.00	324.00	258.94
10	314.00	628.00	321.22	10	314.00	628.00	321.22
11	716.00	1432.00	732.47	11	716.00	1432.00	732.47
12	136.60	273.20	190.41	12	130.00	260.00	181.21
13	90.00	180.00	92.07	13	90.00	180.00	92.07
14	978.00	1956.00	1195.90	14	978.00	1956.00	1195.90
15	676.80	1353.60	608.44	15	690.00	1380.00	620.31
16	22.60	45.20	12.19	16	16.00	32.00	8.63
17	499.20	499.20	178.59	17	500.00	500.00	178.88
18	12.00	96.00	92.13	18	12.00	96.00	92.13
19	30.30	30.30	27.67	19	32.59	32.59	29.76
40	0.00	0.00	0.00	40	3.41	3.41	0.92
Totals	4887.90	9053.90	5164.44		4891.41	9057.41	5164.67

**Table 4.9 Solution Parameters, Test Case 5**

Parameter	Solution 1	Solution 2
Damage Expectancy	0.5406	0.5406
Value Destroyed	5164.44	5164.67
Targets Attacked	4887.9	4891.41
Goal 1 Performance	98.94%	99.44%
Goal 2 Performance	100.00%	100.00%
Goal 3 Performance	100.00%	100.00%
Goal 4 Performance	99.36%	99.31%
Goal 5 Performance	86.20%	86.15%
Goal 6 Performance	99.62%	100.00%
Goal 7 Performance	99.51%	99.89%
Goal 8 Performance	99.32%	99.79%
Goal 9 Performance	100.00%	100.00%
Goal 10 Performance	98.89%	100.00%
Goal 11 Performance	100.00%	100.00%
Goal 12 Performance	100.00%	100.00%
Goal 13 Performance	100.00%	100.00%
Goal 14 Performance	98.06%	100.00%
Goal 15 Performance	92.67%	99.87%
Goal 16 Performance	99.49%	100.00%
Goal 17 Performance	100.00%	100.00%
Goal 18 Performance	100.00%	100.00%
Goal 19 Performance	99.40%	100.00%
Goal 20 Performance	100.00%	100.00%
Goal 21 Performance	4.50%	4.50%
Goal 22 Performance	124.98%	126.41%
Goal 23 Performance	112.21%	115.77%
Goal 24 Performance	116.17%	115.77%
Goal 25 Performance	118.90%	120.35%
Goal 26 Performance	107.12%	108.36%
Goal 27 Performance	98.18%	101.15%
Goal 28 Performance	99.58%	99.23%
Goal 29 Performance	99.08%	100.49%
Goal 30 Performance	90.35%	91.39%
Goal 31 Performance	87.27%	90.08%
Goal 32 Performance	87.13%	86.83%
Goal 33 Performance	71.63%	72.59%
Solution Time (CPU secs)	750	631.2

## **V. Conclusions and Recommendations**

### **Conclusions**

In all of the cases tested, the improved solution methodology maintained damage expectancy and target coverage. The improved solution methodology found the optimal solution in less time than the current methodology in all test cases as well. By removing the 0.9995 adjustment to the right-hand side of the hedge constraints, the improved solution methodology provides the possibility of a truly optimal integer solution. In fact, four of the five test cases provided an optimal integer solution. And in test case four, where an optimal integer solution was not found, there were three and one-half times fewer variables with non-integer values in the improved solution methodology's solution. Finally, the improved solution methodology reduced the number of constraints needed to solve the original LP to optimality by an average of 17%. This resulted in a reduction in size of the basis matrix on average of 30%. Users would benefit greatly by having the improved solution methodology incorporated into the existing AEM.

### **Recommendations**

There are several areas for potential improvement in the improved solution methodology. The AEM can also be improved. This section will outline some of these improvements.

**Implementation in the AEM.** When implementing this methodology into the AEM, some modifications in the algorithm would decrease solution time. After an LP has been solved to optimality, its basis should be used as the starting basis for the next LP. Likewise, any hedge constraints incorporated into the previous LP should remain in the first subproblem of the new LP. This should reduce the number of simplex pivots needed



to solve the new LP subproblem to optimality. To also possibly reduce the solution time on large problems, more than 50 constraints, additional violated constraints can be added to the subproblems to reduce the number of dual simplex pivots required to update the optimal solution. This could incorporate non-binding constraints in the final problem, but will not add redundant constraints.

**Alternate Optimals.** The basic columns are not the only columns that should be kept from one LP to the next. Any columns that have a dual variable equal to zero or near zero should be kept. These columns could provide alternate optimal solutions to the LPs. By exploiting these alternate optimal columns, higher priority hedges may be able to maintain their levels of satisfaction, while not over-constraining lower priorities. As seen with the five test cases, these alternate optimal columns could allow the solver to find truly optimal integer solutions. That is, even if the optimal solution found was not integer, there may exist an equally optimal solution that is. These alternate optimal solutions may also have different goal performances. A fast way to generate alternate optimal solutions might benefit users. One run of the AEM could generate several optimal allocations from the same inputs. From these alternate optimal solutions, users can determine which optimal solution is best for a given scenario. These alternate optimal solutions can be generated after the initial solution procedure by pivoting a column with a zero or near zero dual variable into the current basis. The simplex solver algorithm can then be applied to the last LP with this new basis. This procedure can be done as many times as the user specifies, or until all alternate optimal solutions have been explored.

Exploring all alternate optimals would likely be too time consuming. In the five test cases run, there was an average of 12 columns with zero or near zero dual variables. This implies that there are at least 12 alternate optimal solutions to the original problem. For large problems this number could quickly reach into the hundreds and even possibly the thousands.

## References

1. Best, Michael A, and Klaus Ritter. *Linear Programming: Active Set Analysis and Computer Programs*. Englewood Cliffs, New Jersey 1985.
2. Bozovich, J. F. and others. *The Arsenal Exchange Model (AEM) User's Manual*. Englewood, Colorado: AEM Services, 30 September 1994 (S-94-014-DEN).
3. Cotsworth, William L. *The Arsenal Exchange Model Mathematical Documentation*. Englewood, Colorado: AEM Services, 1991.
4. Faires, Douglas J. and Richard L. Burden. *Numerical Methods*. Boston: PWS Publishing Company, 1993.
5. Gallagher, Mark A. and Elizabeth J. Kelly. "A New Methodology for Military Force Structure Analysis," *OPERATIONS RESEARCH*, 39: 877-885 (November-December 1991).
6. Green Daniel J. *An Integer Solution Heuristic for the Arsenal Exchange Model (AEM)*. MS thesis, AFIT/GST/ENS/94M-04. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1994 (AD-A243923).
7. IMSL, *User's Manual: FORTRAN Subroutines for Mathematical Applications - IMSL Math/Library*. Ver 2.0, September 1991. (MALB-USM-PERFECT-EN9109-2.0)
8. Myers, Danny C. "A Dual Simplex Implementation of a Constraint Selection Algorithm for Linear Programming," *Journal of the Operational Research Society*, 43: 177-180 Feb 1992.
9. Press, William H. and others. *Numerical Recipes: The Art of Scientific Computing*, New York: Cambridge University Press, 1986.
10. Ravidran, A. and others. *Operations Research: Principles and Practice*. New York: John Wiley and Sons, 1987.
11. Winston, Wayne L. *Introduction to Mathematical Programming: Applications and Algorithms*, Boston: PWS-KENT Publishing Company, 1991.

## Vita

Captain Jeffery D. Weir was born on 30 June 1966 in Aurora, Colorado. He graduated from Choctawhatchee High School in Fort Walton Beach, Florida in 1984 and attended the Georgia Institute of Technology in Atlanta, Georgia, graduating with a Bachelor's of Electrical Engineering. He received his commission through the Reserve Officers Training Corps in June 1988. He served his first tour of duty at Malmstrom AFB, Montana. While assigned to Malmstrom AFB, he served as a Deputy Missile Combat Crew Commander, Instructor Deputy Missile Combat Crew Commander, Missile Combat Crew Commander, Instructor Missile Combat Crew Commander, and Senior Instructor Missile Combat Crew Commander. While assigned to Malmstrom AFB, he graduated from Embry Riddle Aeronautical University with a Master's of Aeronautical Science in May 1991. He entered the Graduate School of Engineering, Air Force Institute of Technology, in August 1993.

Permanent Address: 85 Sherrie Lane  
Coates Bend, Al 35901

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 95		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE AN IMPROVED SOLUTION METHODOLOGY FOR THE ARSENAL EXCHANGE MODEL (AEM)			5. FUNDING NUMBERS	
6. AUTHOR(S)  Jeffery D. Weir, Capt, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Air Force Institute of Technology, WPAFB OH 45433-6583			8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GOA/ENS/95M-10	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Air Force Studies and Analysis Agency (USAFSAA) 1570 Air Force Pentagon Washington D.C. 20330-1570			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT  Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The purpose of this research was to design a solution methodology for the Arsenal Exchange Model (AEM) that is faster and contains less precision error than the current one. The current solution methodology modifies some of the original constraints and uses a computationally slow matrix inverter. The improved methodology uses a revised simplex algorithm to first solve a subproblem having only the weapon constraints generated by the AEM. Given this optimal allocation, hedge constraints and target constraints that are violated by the current solution are added to the original subproblem. A dual simplex algorithm is used to find the optimal solution for this new subproblem. By only adding the violated constraints, redundant and identical constraints are not included in any of the subproblems. This eliminates the need to alter the problem as before, and also allows the use of a faster matrix inverter. Additionally, since fewer constraints are used to find the overall optimal solution, fewer computations are necessary. This new methodology was used to solve five test cases. In four of the five test cases, the improved solution methodology produced an optimal integer solution. In all five test cases, it maintained damage expectancy and target coverage, and was better at satisfying the user input goals.				
14. SUBJECT TERMS  Allocations, Mathematical Programming, Goal Programming;			15. NUMBER OF PAGES 51	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT  UL	